Bees and Genetic Algorithms: A Comparison on a Classic Problem

M. Pinninghoff¹, G. Alvarez¹, R. Contreras¹

¹Computer Science Department – University of Concepción
Concepción – Chile
{mpinning,rcontrer}@udec.cl

Abstract. Artificial Bee Colony (ABC) is a meta-heuristic inspired by the process of food seeking of bees, which is used for finding solutions to a great variety of optimization problems. The objective of this work is to implement a bee-based algorithm for solving the CVRP, and to compare the obtained results, with those shown in the benchmark proposed by Augerat, and a previous approach that uses genetic algorithms. Results show an improvement in terms of cost of solutions, validating the approach proposed.

1. Introduction

Product distribution from a depot to different customers is a key issue in an important number of logistic systems management. An adequate planning on how to deal with this problem may impact in important economic savings. Transportation costs are about 20% of the final costs of the goods to be delivered [Toth 2002]. A strategy selection for improving economic results in any organization is often the result of a decision among different alternatives.

The Vehicle Routing Problem (VRP), [Dantzig1959] is an optimization problem that considers the distribution of goods from an initial point of a route (the depot) to different points to be supplied (the customers). If the capacity of every vehicle is the same, the problem evolves to Capacitated Vehicle Routing Problem (CVRP) [Hernandez2004].

Artificial Bee Colony (ABC) is a meta-heuristic inspired by the process of food seeking of bees, used for finding solutions to a great variety of optimization problems. It has shown to be a good mechanism that can obtain results as good as those obtained through classic evolving approaches.

Genetic algorithms (GA) is a known approach for dealing with optimization problems. GA represent a particular class of evolutionary algorithms, used for finding optimal or good solutions by examining only a small part of the possible space of solutions. GA are inspired by Darwin’s theory of evolution. GA are designed to simulate the processes in natural systems necessary for evolution; specifically those that follow the principles of survival of the fittest. As such, they represent an intelligent exploitation of a random search within a defined search space to solve a problem.
The objective of this work is to introduce a bee-based algorithm for solving a classic optimization problem, the CVRP, and to compare the obtained results with those shown in the benchmark proposed by Augerat [Augerat2015] and those obtained by using GA [Pinninghoff2014].

The hypothesis of this work is that ABC obtain better results than those obtained by using AG, because ABC combines a global search with a local search, while GA uses only global search.

This article is structured as follows; the first section is made up of the present introduction; the second section introduces the different definitions and concepts involved in this work; the third section describes the meta-heuristics considered. The fourth section presents the implementation issues and the fifth section shows the results we obtained. The sixth and final section shows the conclusions of the work.

2. The problem

In CVRP, there exists a finite set of customers with different travel cost among them. A specific customer is identified as a depot. Every customer is associated to a specific location and requires to satisfy a specific demand for an unique product. Quantities demanded by customers are previously determined and cannot be partitioned, i.e., every demand has to be satisfied by a vehicle, in a unique delivery operation.

The goal is to find a set of K cycles, every cycle corresponding to the route for a vehicle, having minimum cost. In other words, the solution is a graph containing nodes (the points that are to be satisfied) and edges (that represent different paths to be traversed). The total cost is defined as the sum of the costs of every edge that belong to a cycle. The following constraints are considered: i) every cycle begins and ends in the depot; ii) every customer is visited exactly once by a vehicle \(k\); iii) the sum of customer’s demands of customers belonging to a cycle does not exceed the capacity of vehicle \(k\); iv) the demand of every customer is completely satisfied in each visit.

CVRP is defined through a complete non directed graph \(G=(V,A)\), where the depot is represented by node \(0\). Every feasible route corresponds to paths in \(G\) that begin in node \(0\) and end on node \(0\); the demand for node \(0\) is zero.

Formally, CVRP is defined as follows:

- \(G = (V,A)\), is a complete non directed graph.
- \(V = \{v_0, v_1, v_2, ..., v_n\}\) is a set of nodes, where \(v_0\) is the depot and \(v_1, v_2, ..., v_n\) represent the customers.
- \(A = \{(i,j): i,j \in V, i \neq j\}\) is a set of ordered pairs \((i,j)\), where \(i\) represents the customer in the starting point and \(j\) represents the customer in the final point
- \(C\) is a matrix of costs on the set of edges where \(c_{ij}\) represents the non-negative cost involved in going from the node \(i\) to the node \(j\). This matrix is symmetric and satisfies the triangular inequality, \(c_{ij} \leq c_{ik} + c_{kj}\), for every node in the graph.
- \(K\) is the quantity of vehicles with limited capacity \(Q\).
- \(d_i\) is the known demand for every node \(v_i\).

If we associate binary variables to every edge \((x_{ij}; i, j \in E)\), \(x_{ij} = 1\) if the solution considers the edge \((i,j)\), zero in other situation:
Objective function (minimize):

\[ \min \sum_{i \in V} \sum_{j \in V} (c_{ij}x_{ij}) \]  

Subject to:

\[ \sum_{j \in V} x_{ij} = 1, \forall j \in V \setminus \{0\} \]  

(2)

\[ \sum_{i \in V} x_{ij} = 1, \forall i \in V \setminus \{0\} \]  

(3)

\[ \sum_{i \in V} x_{i,0} = K \]  

(4)

\[ \sum_{j \in V} x_{0,j} = K \]  

(5)

\[ \sum_{i \in S, S \neq \emptyset} \sum_{j \in S} x_{ij} \geq r(S), \forall S \subset V \setminus \{0\} \]  

(6)

\[ x_{ij} \in \{0,1\}, \forall i,j \in V \]  

(7)

where \( r(S) \) represents the lower bound of the number of vehicles that satisfy the set of customers in \( S \). Constraints (2) and (3), represent a condition on the grades of involved nodes: one input and one output for every one of them; constraints (4) and (5) indicate that there exists one input and one output in the depot for every vehicle; constraint (6) avoids the existence of sub-paths; and finally constraint (7) establishes that \( x \) is a binary variable.

3. Meta-heuristics

In the following, there is a general description of the meta-heuristics used in this work: Genetic Algorithms and Artificial Bee Colony Algorithms

3.1. Genetic algorithms (GA)

The structure of a genetic algorithm consists of a simple iterative procedure on a population of genetically different individuals. The phenotypes are evaluated according to a predefined fitness function, the genotypes of the best individuals are copied several times and modified by genetic operators, and the newly obtained genotypes are inserted into the population in place of the old ones. This procedure is continued until a good enough solution is found \([Floreano08]\).

In this work, a chromosome represents a set of paths that shows a possible distribution of vehicles for satisfying every customer. In genetic algorithm terms, each customer is a gene and the identification of a customer represents the value (allele) that this gene has. A good fitness means that a particular routing involves a minimum distance with a minimum number of vehicles. Different genetic operators were considered for this work. These genetic operators are briefly described below:

**Selection:** It is accomplished by using the roulette wheel mechanism \([Floreano08]\). It means that individuals with a best fitness value will have a higher probability to be chosen as parents.

**Cross-over:** It is used to exchange genetic material, allowing part of the genetic information that one individual has, to be combined with part of the genetic information of a different individual. It allows us to combine existing gene values to obtain new and different values, in order to search for better solutions. Due to the nature of the problem we are dealing with, we use a specific cross-over operator: PMX \([Michalewicz2000]\).
Mutation: by using this genetic operator, a slight variation is introduced into the population so that new genetic material is created. It allows us to increase genetic variability.

Some advantages in the use of GA are: i) adaptability, in the sense that in optimization problems any objective function can be used, combined with an arbitrary number of constraints, linear or non-linear, including discrete and continuous variables; ii) robustness, because the use of evolving procedures and operators encourages the search of global instead of local solutions, and iii) flexibility, because GA can be combined with other heuristic procedures, for efficient implementations when solving specific problems [Goldberg1989].

The general GA schema is shown in the following [Karaboga2009].

**Genetic Algorithm:**

1. Initialize Population
2. Evaluation Population
3. repeat
   1. Selection
   2. Crossover
   3. Mutation
   4. Evaluation Population
   5. New Population
4. until requirements are met

3.2. Artificial Bee Colony Algorithms (ABC)

The ABC algorithm is inspired by the behavior of the honeybees in finding nectar sources around the beehives. The emergence of collective intelligence of honeybee swarm consists of three components: food sources, employed foragers and unemployed foragers. This algorithm simulates the behavior of real honeybees considering their foraging behavior. It is used to solve multidimensional and multimodal optimization problems [Karaboga2009].

In the model, the colony of artificial bees consists of three groups of bees: employed bees, onlookers and scouts. Employed bees are responsible for exploiting available food sources and gathering required information about amount of nectar. They share the information with onlookers, and the onlookers select existing food sources to be further explored. When a food source exploited by a bee is exhausted, the bee becomes a scout and starts to search a new food source. A food source is exhausted when the quality of the food source is not improved after performing an allowable number of iterations. The food sources represent solutions of a particular problem and the nectar of a food source is the fitness (evaluation function) of each solution. Furthermore, the model needs a neighborhood operator so that each employee bee finds a new food source.

The ABC algorithm is iterative, and it starts by generating random solutions and assigning each employee bee to a food source. In an iteration step, bees find a new food source close to the previously assigned source, using the neighborhood operator. If the nectar of the new food source is better than the current food source, the new one replaces the former. After all employee bees have finished this process, they share the
information gathered with the onlookers. Then each onlooker selects a food source by using the roulette wheel selection operator [Michalewicz2000]. Each onlooker finds a food source close to its current food source using the neighborhood operator. Then, the best food source, among all the food sources close to the old food source, is selected.

When the employee bees become scouts, they search for new food sources randomly. Then it begins a new iteration and the scout become to be an employee bee again [Szeto2011]. A typical ABC algorithm schema looks like the following [Karaboga2009].

**Artificial Bee Colony Algorithm:**

Initialize Population

repeat

- Place the employed bees on their food sources
- Place the onlooker bees on the food sources depending on their nectar amount
- Send the scouts to the search area for discovering new food sources
- Memorize the best food source found so far

until requirements are met

4. Implementation

Whether we are using AG or ABC, it is necessary to define: i) a representation, i.e., a mechanism for coding problem’s solutions (see Figure 1), ii) an evaluation function for measuring the quality of the solutions (and to compare them), iii) a random initial population generation, to have a variability in the future solutions. Besides that, when dealing with AG, it is necessary to select the genetic operators in charge of the evolution process, to establish the survival criteria for solutions that go to the next generation, and to test the performance of a set of different parameters. Additionally, for ABC, it is necessary to choose the neighborhood operator. These components, when applied to the VRP, are described in the following.

![Figure 1. Codification of a solution](image-url)
When coding a solution, every customer is assigned to an integer number. In Figure 1 (upper part) it is possible to observe a graph showing a solution. The medium part of the figure shows the genetic representation that corresponds to the solution. Paths are determined by adding customer’s demands; a path finishes when a vehicle completes their loading capacity (to ensure a feasible solution). The lower part in Figure 1 shows the codification for the ABC algorithm solution, where zeroes are used to separate different paths. If it is generated a non-feasible solution, it is penalized in the corresponding evaluation function.

In both methods, GA and ABC, evaluation functions differ slightly. For GA this function is equation (1), previously described (see Section 2); the lower is the total cost of the path, the better is the solution.

The evaluation function used in ABC, \(fe(s)\), where \(s\) is a solution, is shown in equation (8):

\[
fe(s) = \text{cost}(s) + \alpha \cdot q(s) \quad (8)
\]

\(\text{cost}(s)\) is the distance traversed for all the involved vehicles, \(q(s)\) represents the value in which the loading capacity \(C\) of a vehicle is exceeded, in the case in which the maximum value is overcome and \(\alpha\) is a parameter that allows penalizing the capacity violation.

Regarding the genetic operators, roulette wheel was chosen for selection, an operator that simulates a roulette wheel in which every chromosome is assigned a roulette area according to the quality of the solution the chromosome represents. This operator selects chromosomes in the population for reproduction. The fitter the chromosome, the more times it is likely to be selected to reproduce, still allowing that, in the random roulette moving, not so good solutions are, with a lower probability, equally chosen [Mitchell1996]. A single crossover operator is not enough for these problems, because there is a high probability of producing non-feasible solutions that may require a complex process for repairing such a solution. This problem explains the use of PMX (Partially Matched Crossover). In partially matched crossover operator two crossover points are selected randomly from the parent’s chromosomes to produce the offspring. The two crossover points give a matching selection, which is used to affect a cross through position-by-position exchange operation [Sivanandam2008]. Mutation operates at the level of the individual. Mutations are small random modifications of the genotype that allow evolution to explore variations of existing solutions. In this work, the mutation operator exchanges the content of two randomly selected positions in the chromosome (genes) [Floreano2008].

Elitism was the selected survival criteria, i.e., a low percentage of chromosomes having the best solutions go to the next generation unmodified; the remaining chromosomes are replaced by new chromosomes obtained after applying the genetic operators.

To establish the best value for the population size, it is necessary to carry out a set of tests taking into account different values. As expected, as the population size increases the processing time also increases, but there is a trade-off if we consider that a
higher size population will have a more important genetic variability, and hence it is possible to require less time (fewer number of iterations) to obtain good solutions.

Following literature recommendations, the crossover percentage is always a high value, typically around 80%.

The mutation probability represents the percentage of individuals that will introduce some non-programmed genetic changes, a process that occurs in nature. The mutation process is not common in an evolving process, and this feature is kept in the GA, setting a mutation percentage that typically is lower than 5%.

The values selected for the considered parameters were obtained after a testing stage and are specified in section Results, in Table 1.

For the ABC method, it is necessary to choose the neighborhood operator and test another set of parameters to obtain promising values. The resulting neighborhood operator was obtained through a combination of two operators: reversing a subsequence and random swaps of reversed subsequences, both have the same probability to be chosen. The first one selects a random length subsequence of the solution and then the order of the corresponding customers (and depot) is reversed. In the second operator, two sequences of customers (and depot) are chosen and swapped, and then each of the swapped subsequences may be reversed with a probability of 50% [Szeto2011].

The ABC parameters considered in this work as described in the following:

Population size, in other words, the number of initial solutions randomly generated. Regarding the number of employee bees and onlooker bees, literature recommends that the number of both types of bee is equal to the number of solutions, although in this work different values were tested, as shown in Table 2. Limit is the number of times an employee bee search for new solutions by applying the neighborhood operator. If the new solution is better than the current solution, this new solution replaces the current one. In [Szeto2011] it is proposed a value that is proportional to the number of customers of the solution, denoted by n. The parameter α is a penalization factor when the capacity of vehicles is exceeded (equation (8)). Finally, it is measured the number of iterations for which the algorithm converges to acceptable solutions.

For all considered parameters, values were obtained after an experimentation stage, and are shown in Table 2, in the next section.

5. Results

For testing the proposal we selected the set of problems presented by Augerat [Augerat2015]. These problems are grouped into three classes (A, B and P). Elements within a class are instances of that class and they exhibit different characteristics: the number of customers and the corresponding demand, the geographic location, the capacity of vehicles (every vehicle in a particular instance exhibits the same capacity), the minimum number of vehicles for solving the problem and the value for the optimal solution (length of the path).

Class A has 27 instances. The geographic location for the customers and their corresponding demands were randomly generated with an uniform distribution;
customers are located around the depot or concentrated in a specific region; the size of instances vary from 31 to 79 customers and the capacity of vehicles is 100. Class B has 23 instances characterized for having the customers grouped into different regions, and as in the case of class A, the depot can be located in the central region; the size of these instances varies from 30 to 77 customers and the capacity of vehicles is, like in the previous set of instances, 100. Class P was not considered because their instances are modified versions of other instances that can be found in the literature.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Population size</th>
<th>Cross-over probability</th>
<th>Mutation probability</th>
<th>Elitism percentage</th>
<th>Number of iterations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value</td>
<td>200</td>
<td>70 – 100%</td>
<td>0 – 20%</td>
<td>10%</td>
<td>68000</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Population size</th>
<th>Employee bees</th>
<th>Onlooker bee</th>
<th>Limit</th>
<th>( \alpha )</th>
<th>Number of iterations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value</td>
<td>25</td>
<td>25</td>
<td>27</td>
<td>54</td>
<td>0.1</td>
<td>65000</td>
</tr>
</tbody>
</table>

Results are shown in the following set of figures. In Figure 2, it is possible to observe results for instances that belong to class A. In Figure 3, it is possible to observe results for instances that belong to class B. In every figure, the y-axis represents the total cost of the solution and the x-axis represents the set of considered instances. Instances are numbered from 1 to 27 for class A, and from 1 to 23 for class B.

Figure 2. Results for instances that belong to class A

For evaluating results we used the Relative Percentage Deviation (RPD), which computes the difference between the evaluation of an optimum solution and the best solution obtained through the use of the meta-heuristics (GA and ABC), according to the formula presented in equation (9).
RPD = \frac{(\text{cost} - \text{optimum})}{\text{optimum}} \times 100 \quad (9)

In (9) “cost” refers to the evaluation obtained by GA or ABC, and optimum is the best value known for the particular instance.

The following figures show a comparison, based on RPD values, for the considered methods (GA and ABC). In every case, the x-axis represents the set of considered instances and the y-axis represents the RPD value. In Figure 4 it is possible to observe the RPD values that correspond to instances that belong to class A and in Figure 5 it is observed the RPD value for instances that belong to class B.

When comparing GA and ABC, results for instances that belong to class A show that from a total of 27 instances, in 15 of them ABC method obtain better results and in 8 instances GA obtains better results; in 4 cases results are similar.
In Figure 4, it is observed that there is a trend to increase the RPD value when the number of customers increases.

![RPD for B instances](image)

**Figure 5. RPD for instances that belong to class B**

Regarding results obtained for instances that belong to class B, for a total of 23 instances, ABC obtains better results in 15 of them, GA obtains better results for 5 instances and for 3 instances results are similar.

Figure 5 shows that in the group of instances that belong to class B, there are negative RPD values. It happens because in the cost definition, it is considered only the distance among customers and not the number of vehicles, and in these cases, solutions require an additional vehicle. The value of RPD for instance 10 is clearly higher than the value of the rest of instances, because the customers’ demands in every group is higher than the capability of a vehicle, and then it is necessary more than on route to satisfy the subset of grouped customers.

Table 3 shows a summary of RPD values for both, A and B classes.

<table>
<thead>
<tr>
<th></th>
<th>Class A</th>
<th></th>
<th>Class B</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>RPD</td>
<td>GA</td>
<td>ABC</td>
</tr>
<tr>
<td>Minimum</td>
<td>0.09</td>
<td>-0.49</td>
<td>-0.89</td>
</tr>
<tr>
<td>Maximum</td>
<td>5.71</td>
<td>4.90</td>
<td>5.54</td>
</tr>
<tr>
<td>Average</td>
<td>2.00</td>
<td>1.79</td>
<td>1.77</td>
</tr>
</tbody>
</table>

In Table 4 and in Table 5 it is shown the influence of the number of customers on the performance of GA and ABC.
In instances having less than 60 customers, the average RPD value for ABC is better than the average RPD value for GA in instances that belong to class A. For instances that belong to class B, the performance is similar. For instances that have more than 60 customers the performance of both methods is similar for instances that belong to class A. However, for instances that belong to class B, the average result is better when using ABC, because ABC performs simultaneously a local and a global search, while GA perform exclusively a global search.

In the group of instances that belong to class B, there are two instances that consider the same number of customers (45 customers). If the performance is depending only on the number of customers, for the ABC it is expected to have a similar RPD in both situations, but RPD values are 0.55 and 5.46 respectively. For GA results are also different, although the difference is not dramatic (0.63 and 2.51 respectively). It occurs that for one of the instances, the sum of demands for a group of customers exceeds the capacity of a vehicle, and hence it is necessary more than one vehicle to satisfy customers that belong to the same group. This fact produces an excessive diversification in the searching process. On the average, the best results for classes A and B are found after 72 seconds.

6. Conclusions

In the meta-heuristics behavior the way in which customers are grouped is important, and it is important the demands customers need to satisfy. The best performance is achieved when customers are grouped into different regions (as in B category instances), and when the number of customers is below 60. ABC presents a slightly better performance when customers are grouped.

From the values obtained it is possible to say that the ABC meta-heuristics represents a proposal that can be considered as a competitive mechanism for solving this particular type of problems.
References